MANUAL Técnico

ANGORA

SISTEMA POS

Developed by KASE

Manual Técnico del Sistema

Web Angora

Organización destino: Fraganc´eys (productora y comercializadora de productos de aseo)

Nombre del Sistema: Angora

Versión del Sistema: 1.0

Tipo de Manual: Manual técnico

Fecha de Elaboración: 15 de agosto de 2025

Área donde fue elaborado: Equipo de desarrollo KASE

Imagen / Portada:

Fraganc'eys
aroma que perdura

Angora

Índice

27. Controles de Auditoría Implementados en el Sistema
28. Glosario de Términos
29. Estándares de Elaboración del Manual

Nota sobre Confección: Los apartados marcados como *"Sección reservada"* serán completados por el autor con sus diagramas/documentos existentes.

Introducción

El sistema web **Angora** es una **plataforma de gestión empresarial integral** creada para **Fraganc´eys**, orientada a **unificar procesos logísticos y financieros en un solo entorno**. Su función principal es **entregar una solución centralizada** que permita administrar de forma eficiente la operación completa de la empresa, reduciendo tiempos, errores y dispersión de información.

Módulos principales (visión general)

- **1.1 Login:** Autenticación de usuarios, recuperación de contraseñas y gestión de sesiones seguras.
- **1.2 Home:** Página de inicio con accesos rápidos, avisos e información de la empresa.
- **1.3 Perfil (Gestión de perfil):** Actualización de datos personales, credenciales y preferencias del usuario.
- **1.4 Dashboard (Resumen de datos):** Indicadores clave de desempeño (KPI) y métricas en tiempo real de finanzas, inventarios y actividad.
- **1.5 Personal:** Gestión de usuarios del sistema por permisos.
- **1.6 Inventario:** Administración de inventario de **productos** y **materias primas**, control por lotes de producción.
- **1.7 Reportes:** Generación de reportes por categoría (finanzas, inventarios y usuarios),

con exportación a Excel.

- **1.8 Ventas:** Emisión de facturas y envío por correo; impresión de tickets de venta.
- **1.9 Clientes:** Portafolio de clientes, datos fiscales y **carteras de crédito**.
- **1.10 Pedidos:** Procesamiento de pedidos, confirmación o rechazo de facturas generadas y coordinación con fabricación.
- **1.11 Proveedores:** Gestión de proveedores y **órdenes de compra** para reabastecimiento de materia prima.

Confección

Nombre del Sistema

Angora

Versión del Sistema

1.0

Tipo de Manual

Manual técnico

Fecha de Elaboración

15 de agosto de 2025

Área donde fue elaborado

Equipo de desarrollo KASE

Introducción

Angora consolida funcionalidades críticas para Fraganc´eys: autenticación, administración de personal y permisos, control de inventario por lotes, facturación y cartera de clientes, pedidos y abastecimiento de materias primas, reportes ejecutivos y operativos. La integración de estos módulos promueve la **trazabilidad de extremo a extremo**, la **consistencia de datos** y la **medición continua** del desempeño mediante un **dashboard** ejecutivo.

Objetivos Generales y Específicos del Sistema

Objetivo general

Desarrollar un sistema POS que permita optimizar y automatizar los procesos administrativos y contables en la empresa Fraganc´eys.

Objetivos específicos

1. Levantar y documentar los requerimientos funcionales y no funcionales del sistema POS mediante entrevistas semiestructuradas con el propietario y análisis de procesos actuales.
2. Diseñar los modelos del sistema POS.
3. Construir el sistema con tecnologías y herramientas modernas.
4. Implementar el sistema POS en la empresa y **medir la efectividad** mediante encuestas y **KPI**.

Normas, Políticas y Procedimientos

- **Normativas y prácticas adoptadas:**
    - **ISO** (alineación general a buenas prácticas de gestión y calidad).
    - **Seguridad informática** (confidencialidad, integridad, disponibilidad; controles de acceso, cifrado de credenciales, auditoría).
    - **Metodologías ágiles** (iteraciones cortas, priorización por valor, demostraciones y retroalimentación continua).
- **Políticas internas:** No aplican políticas internas formales a la fecha.

Definición de las Reglas del Negocio Implementadas

- No se puede **acceder a apartados** sin el **permiso** asociado.
- No se puede **eliminar un cliente con cartera activa**.
- No se puede **registrar productos** sin **materia prima disponible**.
- No se puede **eliminar materia prima**.
- No se puede **eliminar productos.**
- No se puede **confirmar un pedido sin stock**.
- No se puede **registrar usuarios, proveedores o clientes** sin **datos completos**.
- **Validaciones en tiempo real** de todos los procesos: stock, datos, permisos, entre otros.

Fundamentación de la Tecnología Utilizada

- **Backend:** Java con **Spring Boot**.
- **Frontend: React**, HTML, CSS, **Bootstrap**, JavaScript.
- **Base de datos: MySQL**.
- **Control de versiones: Git** y **GitHub**.
- **Herramientas para pruebas de funcionalidad: Postman**

Justificación: Tecnologías **eficaces y potentes**, con ecosistema maduro y amplia comunidad, que permiten una **arquitectura sólida y efectiva**, escalable y mantenible. *(Sin referencias externas solicitadas.)*

Descripción de los Actores del Sistema

- **Administrador:**
    - Accede a todos los módulos; define permisos; gestiona usuarios; supervisa inventarios, ventas, reportes y parámetros del sistema.
    - Nivel de alfabetización digital: medio–alto.
- **Usuario con permisos:**
    - Acceso **condicionado** por permisos; los apartados no autorizados quedan **bloqueados/ocultos en el sidebar**.
    - Realiza tareas operativas (ventas, pedidos, actualización de inventarios, reportes acotados).
    - Nivel de alfabetización digital: medio.

Especificación de Requisitos

Requisitos funcionales

- Gestionar el **personal** de la empresa (administrador o usuarios con permiso).
- **Login** de cuentas de miembros del personal y recuperación de contraseñas.
- Manejo de **inventarios de producto** y **materia prima por lotes**.
- **Reportes** por categoría (finanzas, inventarios, usuarios) con detalles por cada uno y **exportación a Excel**.
- **Facturación** y **envío** de facturas por correo; **impresión de tickets** con totales, ítems, precios, datos del cliente y de la empresa.
- **Portafolio de clientes** con **cartera** y estado crediticio.
- Apartado de **pedidos** con productos a fabricar y posterior **facturación**.
- **Órdenes de compra** a proveedores y envío por correo.

Requisitos no funcionales

- **Interfaz intuitiva** siguiendo principios de usabilidad.
- **Rendimiento:** tiempos de respuesta en carga de datos ≤ **2 segundos** (bajo condiciones nominales).
- **Disponibilidad de datos inmediatos** (consistencia y actualización en tiempo real donde aplique).
- **Confidencialidad y seguridad** de la información.
- **Usabilidad** y accesibilidad razonable.
- **Ligereza:** aplicación web orientada a un uso eficiente de recursos.
- **Escalabilidad** y **mantenibilidad** consideradas en la arquitectura.
- **Seguridad** a base de permisos.

Requisitos de implementación

- **Plataforma web** (navegador moderno).
- **Integración con correo** para envío de facturas y órdenes de compra.
- **Control de versiones** con Git/GitHub.

Requisitos legales y de seguridad

- Protección de datos personales conforme a buenas prácticas.
- Auditoría y trazabilidad de operaciones críticas.

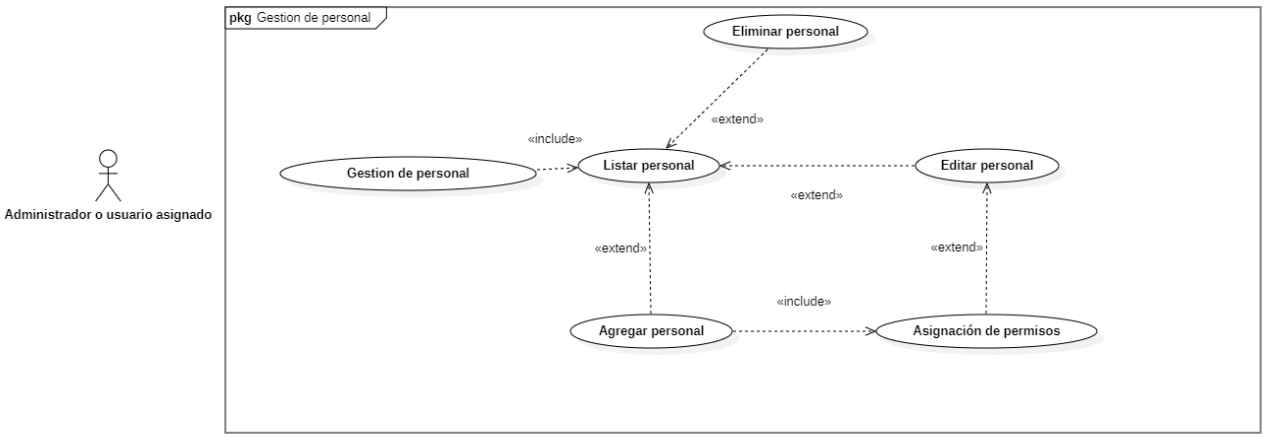
Restricciones en el diseño/implementación

- Acceso y visibilidad **estrictos por permisos**.
- Control de **stock** y **lotes** como ejes de consistencia.

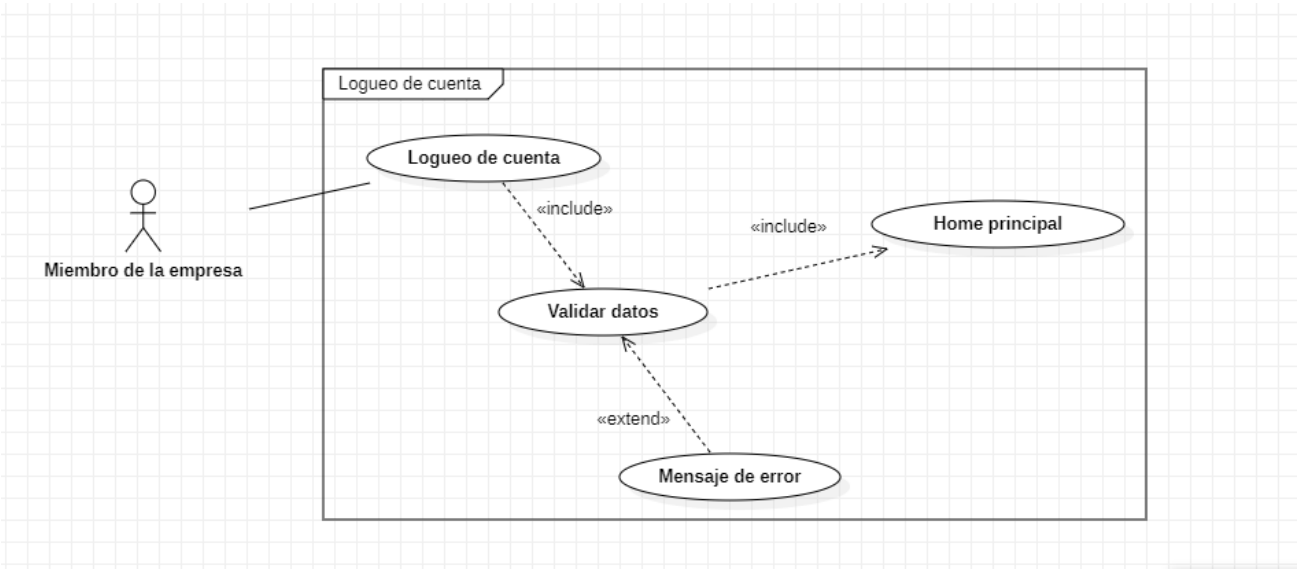
Casos de uso del sistema

Diagramas de casos de uso.

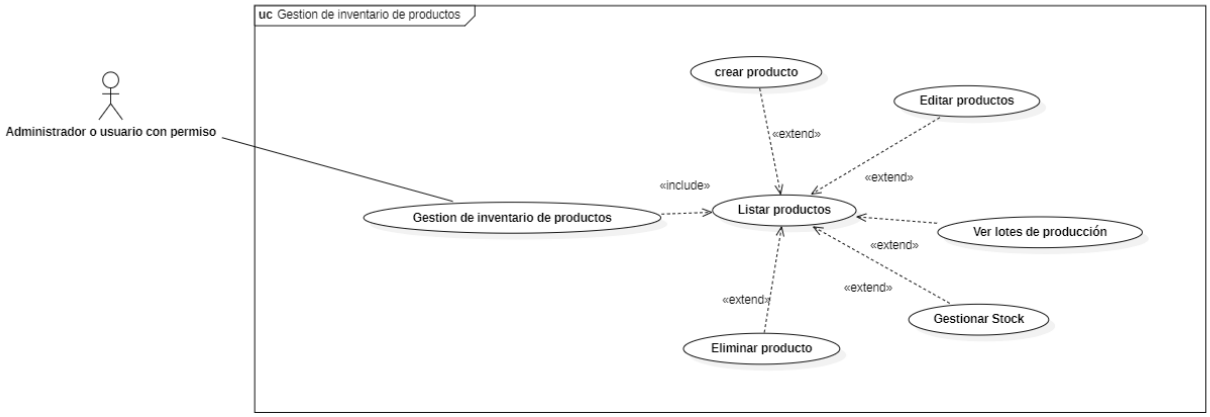
1. **Gestión de personal**



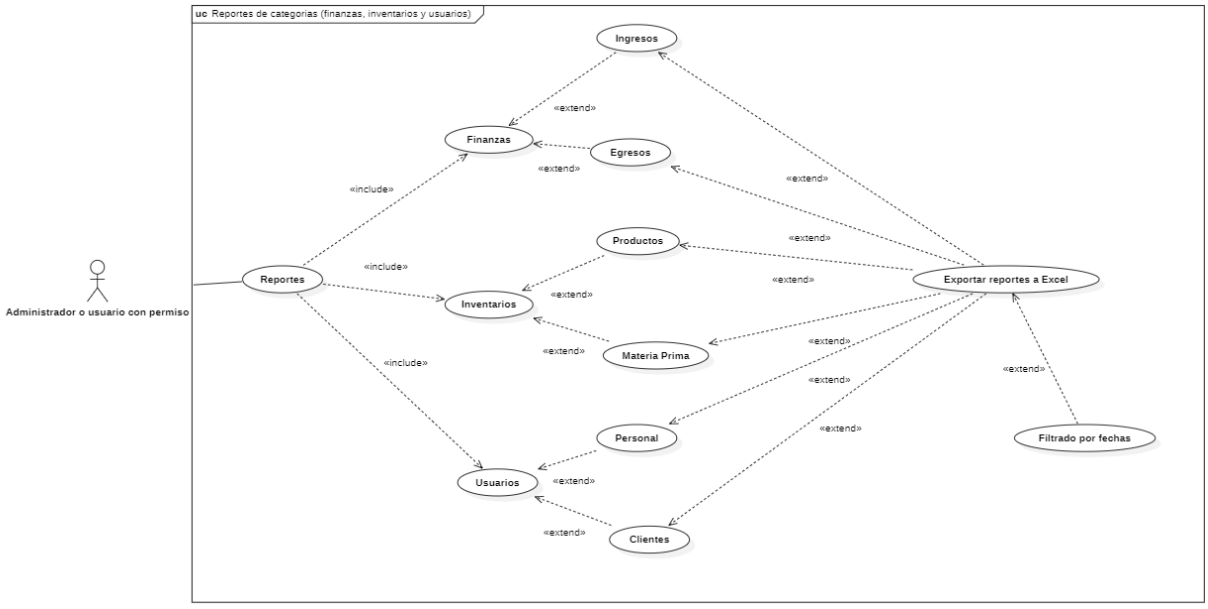
2. Logueo de cuenta



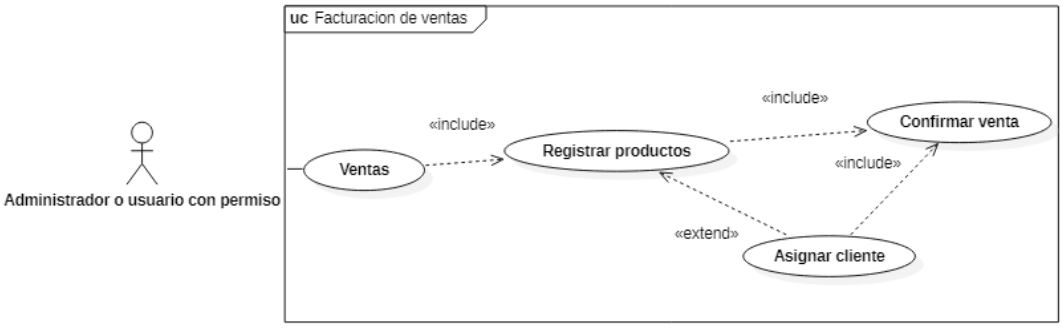
3. Inventario de Productos



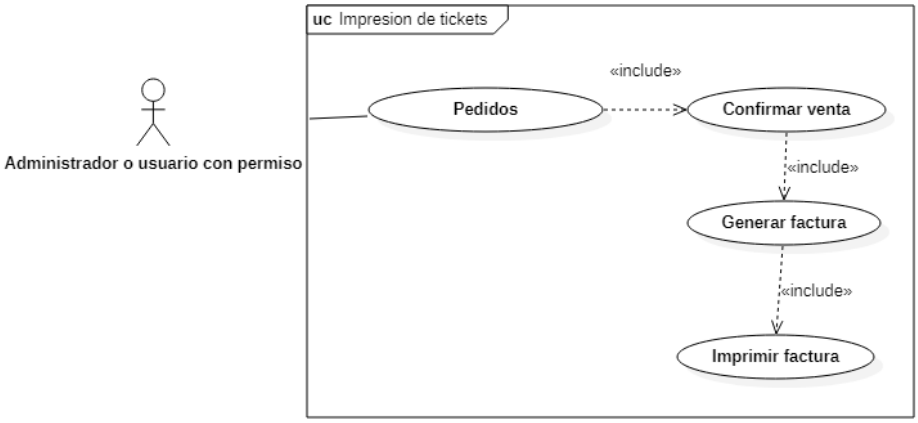
4. Reportes de categorías



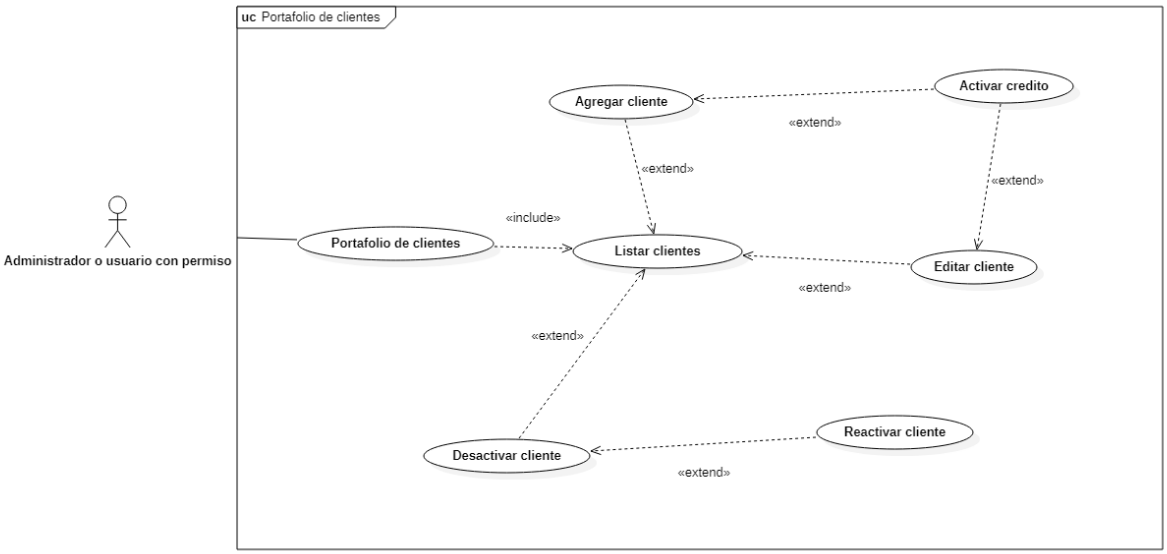
5. Facturación



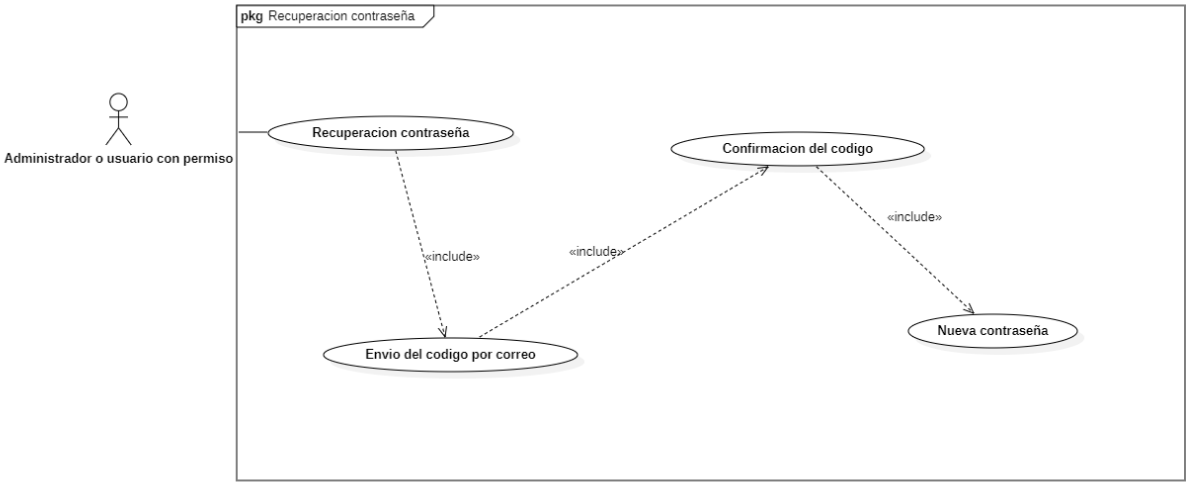
6. Impresión de Ticket



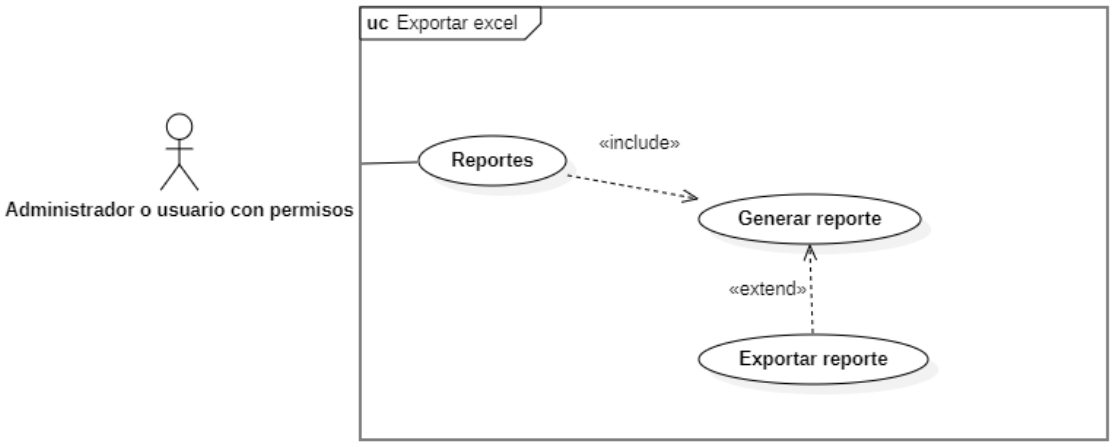
7. Portafolio de clientes



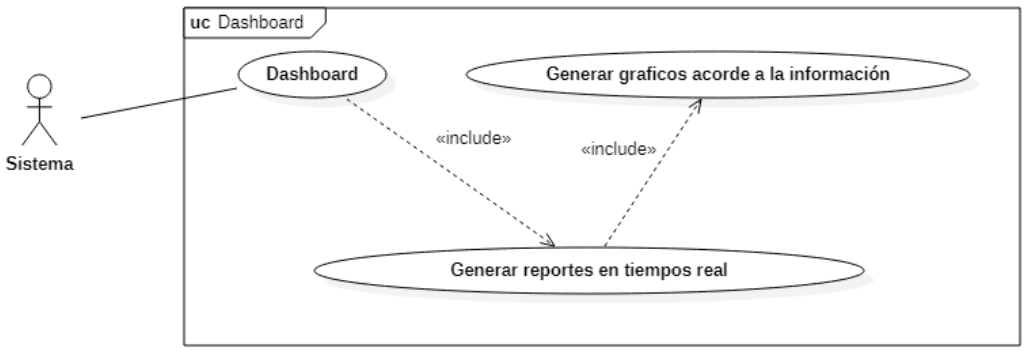
8. Recuperación de contraseñas



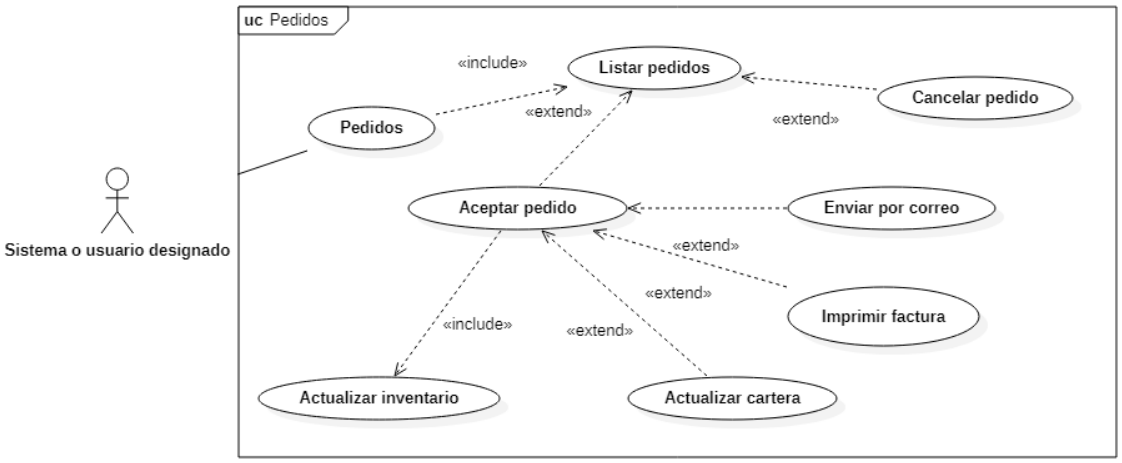
9. Exportar a excel



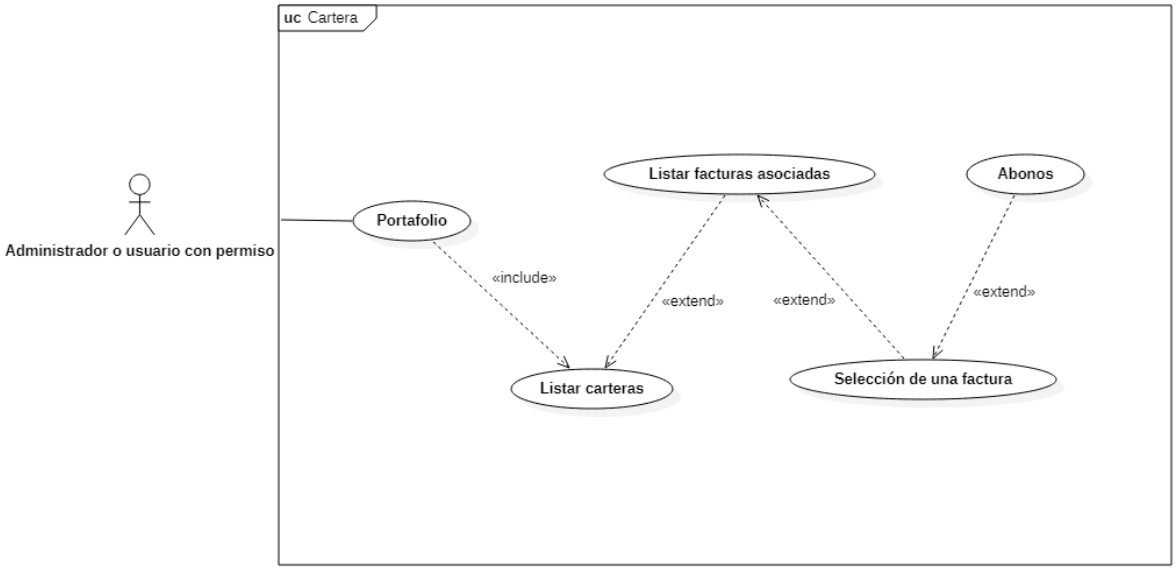
10. Dashboard



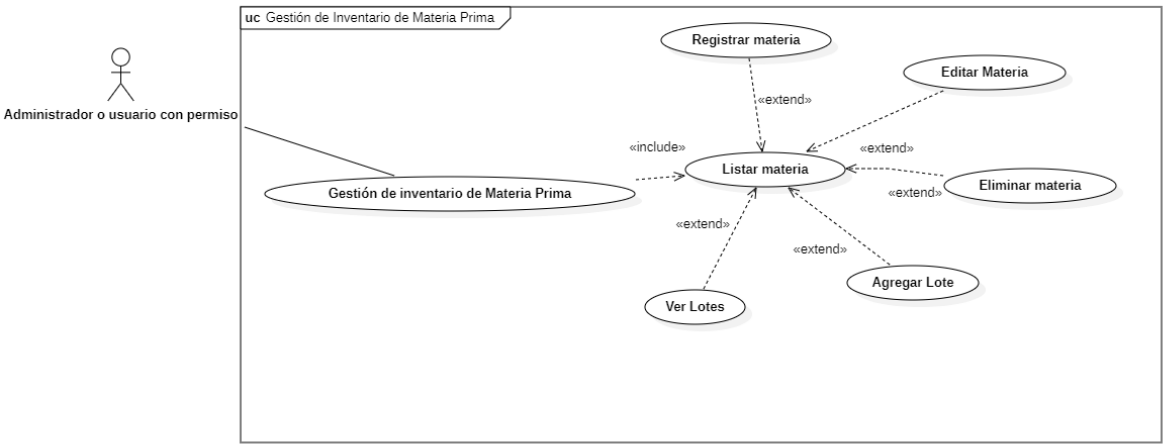
11. Pedidos



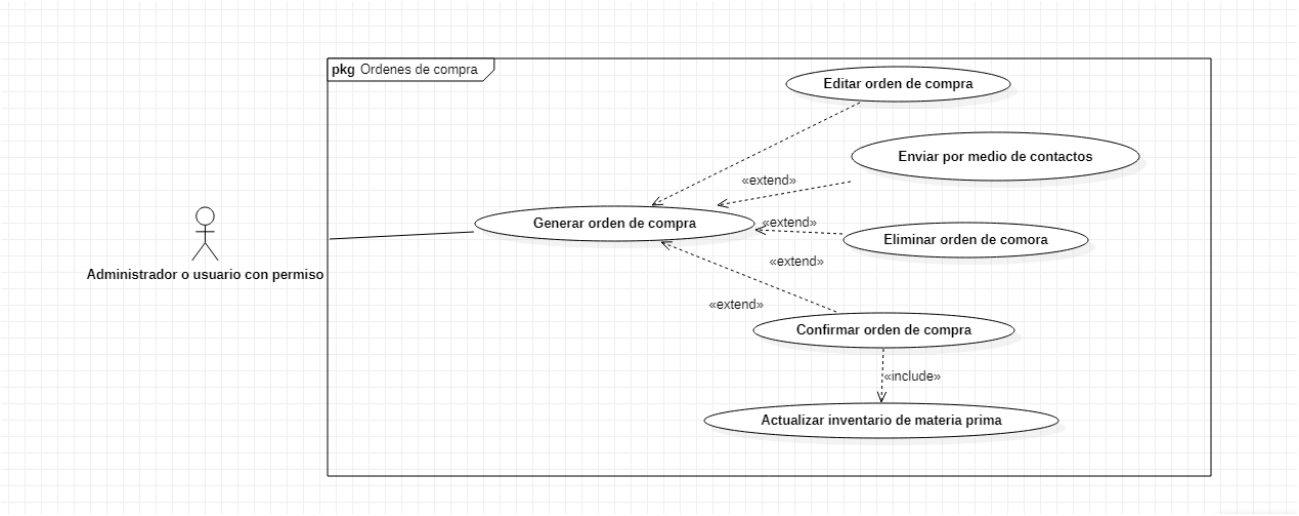
12. Cartera



13. Inventario de materia prima



14. Órdenes de compra



Vista Funcional

2.7.9 Vista Lógica

*El sistema adopta un estilo arquitectónico **MVC (Model-View-Controller)**, que separa la lógica de negocio, la interfaz de usuario y el control de flujos. Esta arquitectura es ideal para aplicaciones web como la presente, permitiendo un desarrollo modular y escalable. El frontend, construido con React, actúa como la capa de Vista, mientras que el backend en Java con Spring Boot implementa el Controlador y el Modelo.*

Patrones de Diseño Implementados

- **MVC:** Como se mencionó, separa la lógica en Modelos (entidades de datos), Vistas (componentes React referente a cada módulo del sistema) y Controladores (rutas en el backend).
- **Builder**: Se utilizó para construir objetos complejos de manera paso a paso, especialmente en la creación de entidades con múltiples atributos. Por ejemplo, se emplea en el backend para construir objetos de tipo usuario que cuentan con múltiples atributos necesarios para el manejo de la seguridad con el módulo de spring security y para armar respuestas API con múltiples parámetros (e.g., un objeto ErrorResponse con mensaje, código y detalles). Esto permite una construcción flexible y legible sin sobrecargar los constructores tradicionales.

2.7.10 Vista de Desarrollo

Estructura de Organización del Código Fuente

El código fuente del sistema está organizado en un repositorio principal alojado en GitHub, accesible en https://github.com/KevinMachado777/Angora, dividido en dos subproyectos principales: frontend (desarrollado con React) y backend (desarrollado con Java y Spring Boot). Esta separación permite un desarrollo paralelo y un despliegue independiente. El control

de versiones se gestiona con Git, utilizando ramas por funcionalidad como feature/inventarios, feature/personal, feature/reportes, etc., y la rama main como principal.

Jerarquía de Paquetes, Carpetas y Módulos

La estructura de carpetas y módulos se organiza de la siguiente manera:

- **root/ (Repositorio Angora)**
    - **frontend/ (Proyecto React)**
        - **node_modules/:** Dependencias instaladas vía npm.
        - **public/:** Archivos estáticos (e.g., index.html).
        - **src/**
            - **api/:** Contiene lógica de autenticación (auth.js) y configuración de Axios (axiosInstance.js).
            - **assets/:** Almacena recursos estáticos, con subcarpetas images (e.g., íconos, fondos) e icons (e.g., SVG).
            - **components/:** Componentes reutilizables (e.g., NotFound.jsx, Button.jsx).
            - **context/:** Maneja el estado global, incluyendo datos del usuario autenticado (e.g., AuthContext.js).
            - **pages/:** Vistas específicas (e.g., NotFound.jsx como página de error).
            - **routes/:** Configuración de rutas (e.g., AppRoutes.js con react-router-dom).
            - **service/:** Servicios personalizados (e.g., userService.js).
            - **styles/:** Hojas de estilo (e.g., notFound.css).
            - **App.jsx:** Punto de entrada de la aplica**ción.**
            - **index.js:** Configuración inicial de React.
        - **package.json:** Dependencias y scripts.

- **angora.app/ (Paquete Java/Spring Boot - Backend)**
  - **src/main/java/com/angora/app/ (Paquetes principales)**
    - **config/:** Configuraciones del sistema, con subcarpeta filter que incluye un validador de token, inicialización de registros, encriptador de contraseñas, configuración de CORS y servicio de imágenes.
    - **contract/:** Interfaces que deben ser implementadas por servicios (e.g., IReporteService.java).
    - **controllers/:** Clases que manejan las solicitudes HTTP (e.g., UserController.java).
    - **entities/:** Clases que representan las entidades de datos (e.g., User.java, Factura.java).
    - **exceptions/:** Excepciones personalizadas (e.g., NotFoundException.java).
    - **repositories/: I**nterfaces de acceso a datos (e.g., UserRepository.java con Spring Data JPA).
    - **services/:** Lógica de negocio (e.g., ClienteService.java).
    - **utils/:** Componentes anotados con @Component (e.g., JwtUtils.java).
  - **src/main/resources/:** Archivos de configuración (e.g., application.properties).
  - **pom.xml:** Dependencias y configuración de Maven.

2.7.11 Vista Física (Despliegue) -> Falta

Sección reservada

*(Este apartado mostrará la **infraestructura de despliegue** del sistema, incluyendo servidores,*

*servicios externos, contenedores, balanceadores y la red que interconecta los componentes. Se podrán incluir **diagramas de despliegue UML** para visualizar la distribución física y las conexiones entre nodos.)*

2.8 Trazabilidad

La trazabilidad del sistema garantiza que cada requisito planteado en las etapas iniciales del proyecto tenga una correspondencia directa con elementos de diseño, implementación, pruebas y despliegue.

Esta sección se representará mediante una **matriz de trazabilidad de requisitos**, donde se establecerán las relaciones entre:

pruebas, para garantizar que todos los requisitos se encuentran cubiertos y correctamente implementados.)

2.9 Requerimientos de Seguridad

El sistema incorpora mecanismos de seguridad para proteger la información de la empresa **Fraganc'eys** y sus usuarios:

1. **Control de acceso por permisos**:
    - Solo usuarios autorizados pueden acceder a secciones específicas del sistema.
    - Los módulos no habilitados no aparecen visibles en el menú lateral (sidebar) para usuarios sin permisos.

2. **Autenticación y sesiones seguras**:
    - Autenticación mediante credenciales únicas para cada usuario.
    - Expiración automática de sesión tras un período de inactividad configurable.

3. **Protección contra eliminación indebida**:
    - No se puede eliminar un cliente con cartera activa.
    - No se puede eliminar materia prima del inventario.
    - No se puede eliminar productos del inventario
    - No se pueden eliminar proveedores

4. **Validaciones de datos**:
    - Validaciones en tiempo real de stock, permisos, datos obligatorios y consistencia de información.
    - Restricción para confirmar pedidos sin stock disponible.

5. **Transmisión segura de datos**:
    - Uso de **HTTPS** para el cifrado de la comunicación entre el frontend y el backend.

2.10 Plan de Pruebas

El sistema será sometido a pruebas exhaustivas para garantizar la calidad del producto final, siguiendo la metodología de pruebas funcionales y no funcionales.

Tipos de pruebas:

- **Pruebas unitarias**: Validar el correcto funcionamiento de cada componente individual.
- **Pruebas de integración**: Verificar la correcta interacción entre módulos del sistema.
- **Pruebas funcionales**: Confirmar que cada requerimiento funcional se cumple según lo documentado.
- **Pruebas de rendimiento**: Asegurar que el tiempo de respuesta no supere los 2 segundos para operaciones críticas.

- **Pruebas de seguridad**: Validar que no existan vulnerabilidades de acceso no autorizado.
- **Pruebas de aceptación**: Realizadas junto con el cliente para confirmar que el sistema cumple sus expectativas.

Criterios de aceptación:

- Todos los casos de prueba deben ejecutarse sin errores críticos.
  Cumplimiento del 100% de los requerimientos funcionales acordados.

2.11 Despliegue e Implementación

El despliegue del sistema POS se realizará de la siguiente manera:

1. **Preparación del entorno**:
    - Configuración del servidor de backend con Java Spring Boot.
    - Configuración de la base de datos MySQL.
    - Configuración del servidor web para React (frontend).

2. **Instalación y configuración**:
    - Clonación del repositorio desde GitHub.
    - Instalación de dependencias en frontend y backend.
    - Configuración de variables de entorno para conexión con la base de datos y API.

3. **Migración de datos iniciales**:
    - Carga de un usuario(admin) por defecto.
4. **Puesta en marcha**:
    - Pruebas en entorno de producción.
    - Capacitación del personal en el uso del sistema.

2.12 Mantenimiento y Soporte

El mantenimiento del sistema será **correctivo** y **evolutivo**:

- **Mantenimiento correctivo**: Resolución de errores detectados tras la puesta en marcha.
- **Mantenimiento evolutivo**: Incorporación de nuevas funcionalidades solicitadas por el cliente o necesarias para la mejora continua.
- **Mantenimiento preventivo**: Optimización de código, limpieza de datos innecesarios y mejoras de seguridad.

El soporte técnico incluirá:

- Canales de contacto vía correo electrónico.
- Respuesta prioritaria a fallos críticos.

3. Arquitectura del Sistema

3.1 Descripción General de la Arquitectura

El sistema web **Angora** adopta una arquitectura **cliente-servidor** basada en un enfoque **multicapa** (frontend, backend y base de datos).

- **Frontend**: Desarrollado con **React.js**, responsable de la interacción con el usuario.
- **Backend**: Desarrollado con **Java Spring Boot**, encargado de la lógica de negocio y la comunicación con la base de datos.
- **Base de Datos**: Implementada en **MySQL**, con un modelo relacional optimizado para

integridad referencial y consistencia de datos.

3.2 Vista de Desarrollo (Vista de Componentes)

Esta vista describe cómo se organizan los módulos y componentes del sistema en el entorno de desarrollo.

- **Frontend**:
    - Componentes reutilizables para formularios, tablas y modales.
    - Integración con API REST usando Axios.
    - Manejo de estado mediante Hooks y Context API.

- **Backend**:
    - Controladores REST (Controllers) para recibir y responder solicitudes.
    - Servicios (Services) que implementan la lógica de negocio.
    - Repositorios (Repositories) para la persistencia de datos.

- Entidades (Entities) que mapean las tablas de la base de datos.

3.4 Vista Física (Despliegue) -> Falta

La vista física describe la infraestructura donde el sistema se ejecuta:

- **Servidor Backend**: Java Spring Boot, hospedado en un servidor local o nube.
- **Servidor Frontend**: React.js compilado y servido mediante Nginx o Apache.
- **Base de Datos**: MySQL, alojada en el mismo servidor o en un servidor independiente para mayor escalabilidad.

Sección reservada *(Diagrama de despliegue con nodos y conexiones de red).*

3.5 Vista Lógica

Describe la organización del sistema desde la perspectiva del diseño orientado a objetos:

- **Estilo arquitectónico**: multicapa (presentación, negocio y datos).

- **Patrones de diseño utilizados**:
  - **MVC** (Model-View-Controller) adaptado para backend.
  - **DAO** (Data Access Object) para la persistencia.
  - **Singleton** en la configuración de conexión a base de datos.

- **Mecanismos de diseño**:
  - Validación de datos en backend y frontend.
  - Uso de DTOs (Data Transfer Objects) para transferir datos entre capas.

Diagrama de clases

Diagrama de clases.pdf

5. Reglas de Negocio

Las reglas de negocio de Angora definen las condiciones, restricciones y políticas que rigen el funcionamiento del sistema, asegurando que las operaciones se realicen conforme a los lineamientos de la empresa y la normativa vigente.

5.1 Reglas Generales

- ***RN-001 – Autenticación Segura**: Todo usuario debe iniciar sesión con credenciales válidas; se aplicará cifrado de contraseñas y expiración de sesiones inactivas.*

- ***RN-002 – Control de Acceso por permisos**: Las funcionalidades se habilitan o restringen según el permiso asignado.*

- ***RN-003 – Disponibilidad del Sistema**: El sistema debe estar operativo 24/7, salvo mantenimientos programados notificados con antelación.*

5.2 Reglas de Negocio por Módulos importantes

5.2.1 Módulo de Pedidos

- ***RN-PED-001 – Estado Inicial**: Todo pedido nuevo se registra como "Pendiente".*
- ***RN-PED-002 – Disponibilidad de Stock**: No se permite confirmar un pedido si la cantidad solicitada supera el stock disponible.*
- ***RN-PED-003 – Cálculo Automático**: El sistema calcula subtotal, impuestos y total*

automáticamente.

5.2.2 Módulo de Facturación

- ***RN-FAC-001 – Confirmación Obligatoria**: Las facturas deben ser confirmadas por un usuario autorizado antes de enviarse al cliente.*
- ***RN-FAC-002 – Formato PDF**: Toda factura se genera en formato PDF con folio único y firma digital.*
- ***RN-FAC-003 – Envío Automático**: Una vez confirmada, la factura se envía por correo electrónico al cliente.*

5.2.3 Módulo de Órdenes de Compra

- ***RN-OC-001 – Relación con Proveedor**: No se puede generar una orden de compra sin un proveedor asociado registrado en el sistema.*

  ***RN-OC-002 – Actualización de Costos**: El costo unitario registrado debe corresponder a la última negociación registrada con el proveedor.*
- ***RN-OC-003 – Estado Inicial**: Toda orden nueva queda como "En Proceso" hasta su recepción.*

5.2.4 Módulo de Inventario

- ***RN-INV-001 – Control de Existencias**: El inventario no puede registrar cantidades negativas.*
- ***RN-INV-002 – Movimientos Registrados**: Todo ingreso o egreso debe generar un movimiento con fecha, cantidad y usuario responsable.*
- ***RN-INV-003 – Actualización Automática**: El inventario se actualiza automáticamente al confirmar facturas o recepciones de órdenes de compra.*

5.3 Cumplimiento Legal y Normativo

- *Cumplimiento de normativas fiscales nacionales.*
- *Protección de datos personales conforme a la legislación vigente (Ley 1581 de 2012 en Colombia).*
- *Conservación de registros electrónicos por el tiempo estipulado por la ley tributaria.*

6. Requisitos Funcionales y No Funcionales

*En este capítulo se detallan los requisitos que definen el comportamiento y las condiciones del sistema Angora. Los **requisitos funcionales** describen lo que el sistema debe hacer, mientras que los **no funcionales** establecen las restricciones de calidad, rendimiento y seguridad.*

6.1 Requisitos Funcionales (RF)

ID	*Nombre*	*Descripción*	*Prioridad*	*Módulo*
RF-001	*Gestión de personal*	*Permite tener un manejo sobre los miembros de la empresa tanto para administrarles funcionalidades dentro del sitio o modificaciones que se le puedan hacer a los mismos.*	*Alta*	*Personal*
RF-	*Logueo de cuentas*	*Permite loguear un administrador o usuario para acceder al sistema*	*Alta*	*Login*

002				
RF-003	*Inventario de productos*	*Almacena los productos en la base de datos, al igual que editar su información, eliminar los productos y llevar una trazabilidad de los productos que salen y entran del inventario.*	*Alta*	*Inventarios*
RF-004	*Reportes por categoría*	*Permite obtener reportes acertados según categoría, para que el administrador o usuario con permiso sepa el estado de la empresa*	*Media*	*Reportes*
RF-005	*Facturación*	*Permite ver a detalle los datos del producto, datos del usuario y totales de la venta*	*Alta*	*Ventas*
RF-006	*Impresión de tickets*	*Permite imprimir los datos de compra para el control de ventas y constancia de la compra .*	*Media*	*Ventas*
RF-007	*Portafolio de clientes*	*Se creará un apartado de clientes para el registro, edición o desactivación de los clientes de la empresa, al igual que el apartado de cartera para un seguimiento del crédito del mismo*	*Alta*	*Clientes*

RF-008	*Recuperación de contraseñas*	*Permite que un usuario recupere su contraseña en caso de haberla olvidado*	*Media*	*Login*
RF-009	*Exportación a excel*	*Permite que la empresa pueda exportar los reportes generados por el sistema a una plantilla Excel.*	*Media*	*Inventarios/ Reportes*
RF-010	*Dashboard*	*Permite que los usuarios, tengan un resumen gráfico del estado financiero y logístico de la empresa.*	*Baja*	*Dashboard*
RF-011	*Pedidos*	*Apartado en el que se verán los pedidos que se hagan en la empresa*	*Alta*	*Pedidos*
RF-012	*Cartera*	*Permite que la empresa lleve el historial crediticio y en qué estado se encuentra cada cliente con la empresa.*	*Alta*	*Clientes*
RF-013	*Inventario de materias primas*	*Almacena la materia prima en la base de datos utilizando lotes, al igual que editar su información y eliminar los que prefiera.*	*Alta*	*Inventarios*
RF-014	*Gestión de órdenes de compra*	*Se van a crear las órdenes de la materia prima que se necesite al respectivo proveedor, así como el*	*Alta*	*Proveedores*

		registro de la misma dentro del inventario de materia prima	

6.2 Requisitos No Funcionales (RNF)

ID	*Nombre*	*Descripción*	*Prioridad*
RNF-001	*Interfaz intuitiva*	*La interfaz de la página debe tener un diseño claro, con un acomodamiento consistente de los elementos de la interfaz (botones, menús, campos de entrada) en todas las páginas del sistema.*	*Alta*
RNF-002	*Rendimiento*	*Las operaciones críticas deberán ejecutarse en menos de 3 segundos en condiciones normales.*	*Alta*
RNF-003	*Sin límite en la DB*	*Se tendrá un equipo central potente y una capacidad de base de datos amplia.*	*Alta*
RNF-004	*Reportar datos inmediatos*	*Las operaciones necesarias para los reportes se harán de manera inmediata a la hora de interactuar con algún apartado (Facturación e inventario).*	*Media*
RNF-005	*Confidencialidad*	*El sistema tendrá las mejores prácticas de seguridad y confidencialidad.*	*Alta*

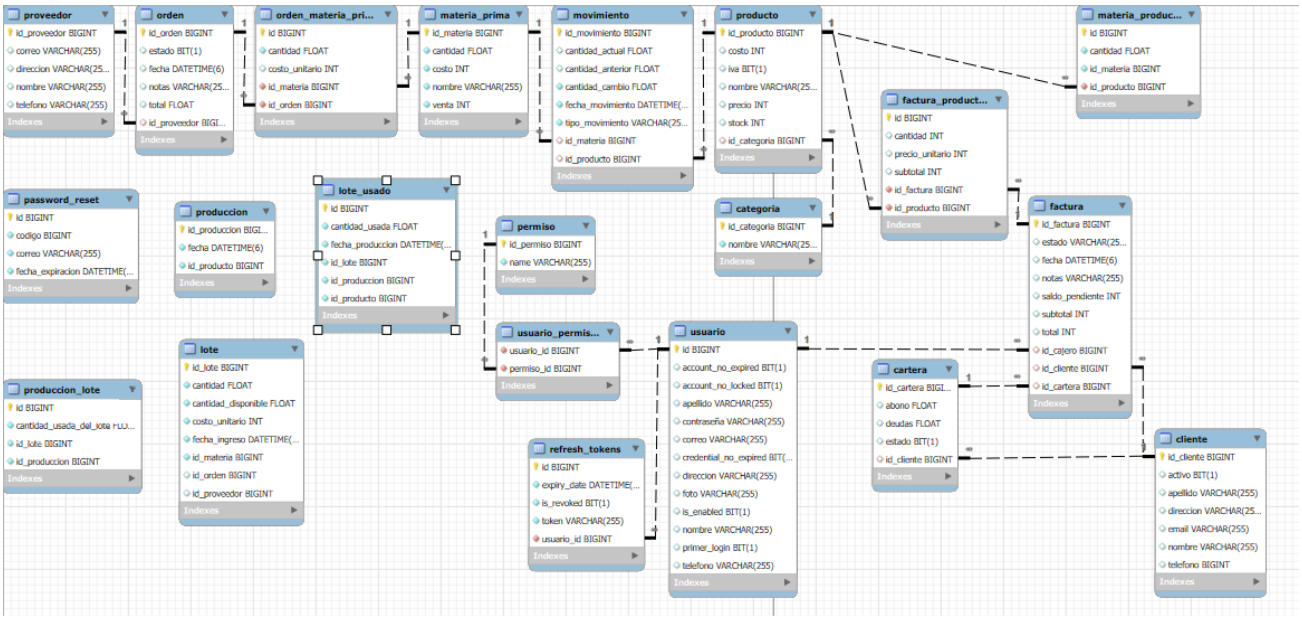
RNF-006	*Usabilidad del software*	*El usuario podrá interactuar con el sistema con apartados interactivos y muy bien marcados, además, será en su totalidad responsivo.*	*Media*
RNF-007	*Seguridad del software*	*Los usuarios del sistema tendrán un acceso controlado a los apartados según los permisos asignados por el administrador o personal asignado.*	*Alta*

---

7. Modelo de Datos

En este capítulo se describen los elementos y relaciones que conforman el modelo de datos del sistema Angora. Se presentará el ***diagrama entidad-relación (ER)****, las descripciones de las tablas, así como las claves primarias, claves foráneas, restricciones y relaciones cardinales entre entidades.*

7.1 Diagrama Entidad-Relación (ER)



7.2 Descripción de Entidades y Atributos

A continuación se detallan las entidades y sus atributos

Diccionario de datos

---

8. Diseño de la Interfaz de Usuario

En este capítulo se documenta el diseño visual y funcional de la interfaz de usuario del sistema Angora, detallando la distribución de elementos, flujos de navegación y características de accesibilidad

.*Manual de usuario*

8.1 Principios de Diseño Adoptados

El diseño de la interfaz de usuario sigue los siguientes principios:

- ***Consistencia visual y funcional****: uso coherente de colores, tipografías, iconografía y estilos de botones.*
- ***Usabilidad****: priorización de la facilidad de uso para minimizar la curva de aprendizaje.*
- ***Accesibilidad****: cumplimiento de criterios básicos de accesibilidad para usuarios con necesidades especiales.*
- ***Compatibilidad multiplataforma****: adaptación visual y funcional a diferentes dispositivos y resoluciones de pantalla (diseño responsivo).*

8.2 Mapa de Navegación del Sistema

Mapa de Navegación

8.3 Prototipos y Mockups

- Prototipo de bajo nivel: Mockup Balsamiq
- Prototipo de alto nivel: Figma Angora

8.4 Descripción de Vistas Principales

A continuación, se listan las principales vistas de Angora con una breve descripción de su funcionalidad:

- ***Login****: pantalla inicial para autenticación de usuarios mediante credenciales (usuario/contraseña).*
- ***Home****: panel principal que muestra accesos rápidos y datos de la empresa*
- *.**Dashboard:** módulo que muestra reportes detallados diarios, semanales o mensuales.*

- ***Perfil****: módulo para gestión de datos personales.*
- ***Personal:*** *módulo donde se van a gestionar todos los usuarios del sistema.*
- ***Inventarios****: control de productos, materias primas, lotes de producción con actualización en tiempo real a medida que se mueva el sistema.*
- ***Reportes****: generación y exportación de reportes de inventarios, financieros y logísticos a excel.*
- ***Ventas****: módulo donde se hacen las ventas de productos a clientes registrados o consumidores finales.*
- ***Clientes:*** *módulo donde se manejan los clientes y sus estados de crediticios con la empresa.*
- ***Pedidos****: visualización y confirmación de pedidos pendientes, con opción de generar facturas.*
- ***Proveedores y órdenes de compra****: creación y edición de proveedores a los que se le hacen ordenes de compras únicamente de materia prima para abastecimiento de ese inventario.*

8.5 Accesibilidad y Responsividad

El sistema asegura una correcta visualización en dispositivos móviles, tabletas y monitores mediante el uso estratégico de media queries, que ajusta dinámicamente el layout y los tamaños de los elementos según el ancho de pantalla. Por ejemplo, en pantallas menores a 860px, el diseño se adapta a un formato vertical para optimizar el espacio, mientras que en resoluciones por debajo de 600px y 400px se reducen proporcionalmente el tamaño del texto, los componentes y los elementos gráficos, asegurando que todo permanezca legible y bien distribuido. Esta aproximación garantiza una experiencia fluida sin necesidad de zoom, adaptándose seamlessly a una amplia variedad de dispositivos, desde teléfonos pequeños hasta pantallas de escritorio de alta resolución, con transiciones suaves que preservan la

usabilidad en cada contexto.

9. Seguridad del Sistema

Este capítulo describe los mecanismos implementados en Angora para garantizar la seguridad de la información, la integridad de los datos y la protección frente a accesos no autorizados.

9.1 Autenticación de Usuarios

El sistema implementa un mecanismo de autenticación basado en JWT (JSON Web Token) y Spring Security, donde los usuarios inician sesión proporcionando su correo y contraseña a través de un endpoint protegido. Al crearse un usuario, se genera una contraseña temporal que se envía al correo registrado, y al primer inicio de sesión, el sistema obliga al usuario a cambiar esta contraseña; una vez cambiada, la sesión permanece activa con el nuevo token JWT y el refresh token, permitiendo un uso continuo sin interrupciones. El token JWT, firmado con una clave secreta, se incluye en el encabezado Authorization (e.g., Bearer), mientras que el refresh token, almacenado en HTTP-only cookies, renueva el JWT al expirar a través de un endpoint dedicado validado por Spring Security. El cierre de sesión implica eliminar ambos tokens localmente, con la opción de revocar el refresh token en el backend para mayor seguridad. Para la recuperación de contraseña, el usuario solicita un código de 6 dígitos enviado al correo, con un límite de validez de 3 minutos; si el código se ingresa correctamente dentro de este plazo, el usuario puede cambiar su contraseña, de lo contrario, el proceso debe reiniciarse. Este enfoque asegura autenticación segura, manejo inicial de contraseñas y recuperación efectiva.

9.2 Autorización y Control de Acceso

*El sistema implementa un mecanismo de autorización y control de acceso basado en permisos utilizando Spring Security, sin emplear roles, donde los permisos se asignan directamente a usuarios para acceder a funciones específicas. La estructura de permisos está definida mediante una lista de autoridades (e.g., CLIENTES, VENTAS, INVENTARIOS, PERSONAL, REPORTES, PROVEEDORES, PEDIDOS, DASHBOARD) configuradas en la clase SecurityConfig dentro del paquete Angora.app.Config. Estos permisos se asignan a endpoints específicos a través de métodos HTTP, controlando el acceso a módulos como clientes, carteras, inventario, personal, reportes, proveedores, pedidos, y dashboard. Por ejemplo, las rutas /clientes/** requieren permisos como CLIENTES o VENTAS para operaciones GET, mientras que /inventarioProducto/** exige INVENTARIOS para acciones como POST, PUT y DELETE. El acceso a funciones críticas, como la eliminación de lotes (/lotes/** con denyAll) o la confirmación de órdenes (/ordenes/confirmar/** con PROVEEDORES), se restringe explícitamente mediante hasAuthority o denyAll en HttpSecurity. El filtro JwtTokenValidator, integrado antes de UsernamePasswordAuthenticationFilter, valida el token JWT en cada solicitud, y DaoAuthenticationProvider asegura que los permisos asociados al usuario, almacenados en la base de datos y recuperados por UserDetailService, sean verificados, denegando el acceso a cualquier endpoint no autorizado con anyRequest().denyAll().*

9.3 Cifrado y Protección de Datos

El sistema emplea cifrado robusto para proteger datos sensibles y asegurar su transmisión. Las contraseñas de los usuarios se almacenan cifradas en la base de datos utilizando BCryptPasswordEncoder, integrado en Spring Security a través del bean PasswordEncoder, que aplica un algoritmo de hash unidireccional con salt para prevenir ataques de fuerza bruta. Datos sensibles, como tokens JWT y detalles de usuarios, se manejan en memoria y se transmiten mediante HTTPS con TLS 1.2 o superior, garantizando encriptación de extremo a

extremo entre el cliente y el servidor; esto se configura a nivel de infraestructura (e.g., servidores Tomcat o contenedores Docker con certificados SSL). Además, la clase JwtUtils genera y valida tokens JWT con una clave secreta fuerte, protegiendo la integridad y confidencialidad de los datos de autenticación. La configuración de CORS en SecurityConfig permite solo orígenes confiables (e.g., http://localhost:5173), y los filtros como JwtTokenValidator aseguran que los datos sensibles en las solicitudes estén protegidos, minimizando riesgos de exposición.

9.4 Prevención de Vulnerabilidades Comunes

- *El sistema Angora implementa prácticas de seguridad para mitigar riesgos comunes de vulnerabilidades. Para prevenir SQL Injection, se utilizan consultas parametrizadas y un ORM seguro como Spring Data JPA, que abstrae las interacciones con la base de datos y evita la inyección de código malicioso mediante la validación automática de entradas. Contra Cross-Site Scripting (XSS), se aplica sanitización y validación de datos de entrada en los controladores de Spring, asegurando que los datos recibidos (e.g., parámetros de solicitudes HTTP) se limpien antes de procesarse o renderizarse, junto con la desactivación de CSRF en SecurityConfig para APIs stateless con JWT. Para mitigar Cross-Site Request Forgery (CSRF), aunque CSRF está desactivado (csrf.disable()), se valida el origen de las solicitudes mediante la configuración de CORS en SecurityConfig, permitiendo solo orígenes confiables (e.g., http://localhost:5173). Finalmente, para evitar la exposición de información sensible, se ocultan datos críticos como tokens JWT y detalles de usuarios en logs y mensajes de error, configurando Spring Security para devolver respuestas genéricas en caso de fallos de autenticación, protegiendo así contra revelaciones accidentales.*

---

10. Plan de Pruebas

Este capítulo define la estrategia y metodología para verificar que el sistema Angora cumpla con los requisitos funcionales y no funcionales establecidos, asegurando su correcto funcionamiento antes, durante y después de su implementación.

10.1 Objetivos de las Pruebas

- *Garantizar que todas las funcionalidades del sistema operen conforme a las especificaciones.*
- *Detectar y corregir errores antes de la entrega al cliente final.*
- *Validar el rendimiento, seguridad y usabilidad del sistema.*
- *Asegurar la integración correcta entre los diferentes módulos.*

10.2 Tipos de Pruebas Utilizadas

- ***Pruebas Unitarias**: verifican el correcto funcionamiento de componentes individuales (funciones, clases, métodos).*
- ***Pruebas de Integración**: validan la interacción entre módulos y servicios.*
- ***Pruebas Funcionales**: confirman que el sistema cumple con los requisitos funcionales definidos.*
- ***Pruebas de Rendimiento**: miden tiempos de respuesta, consumo de recursos y capacidad de carga.*
- ***Pruebas de Seguridad**: verifican la resistencia del sistema a ataques y accesos no autorizados.*
- ***Pruebas de Usabilidad**: evalúan la experiencia del usuario y la facilidad de navegación.*

10.3 Herramientas de Pruebas

El sistema utiliza exclusivamente Postman como herramienta de pruebas, empleándose para validar los endpoints de la API, probar flujos de autenticación con JWT, verificar respuestas de solicitudes HTTP (e.g., GET, POST, PUT, DELETE), y realizar pruebas de integración entre el frontend y el backend. Postman se configura con colecciones para organizar las pruebas por módulos (e.g., clientes, inventario, autenticación) y se utiliza para simular escenarios como el envío de correos con códigos de recuperación o la generación de tokens.

10.4 Criterios de Aceptación

- *Todos los casos de prueba deben pasar sin errores críticos.*
- *No deben existir fallos de seguridad detectados.*
- *El rendimiento debe estar dentro de los parámetros acordados.*
- *La interfaz debe cumplir con los estándares de usabilidad definidos.*

10.5 Casos de Prueba -> Falta

Sección reservada

(Incluir tabla con los casos de prueba, describiendo: ID, módulo, descripción, pasos para reproducir, datos de prueba, resultado esperado, resultado obtenido y estado.)

---

11. Plan de Implementación

Este capítulo describe la estrategia para poner en marcha el sistema Angora en el entorno de producción, asegurando una transición controlada y minimizando riesgos operativos.

11.1 Objetivos de la Implementación

- *Asegurar que el sistema esté disponible para todos los usuarios finales en el plazo establecido.*
- *Minimizar el tiempo de inactividad durante el despliegue.*
- *Garantizar la integridad de los datos durante la migración.*
- *Reducir riesgos asociados a fallos en el arranque.*

11.2 Estrategia de Implementación -> Falta

Sección reservada

(Describir si la implementación será Big Bang, incremental, por fases, en paralelo, o piloto. Indicar justificación de la estrategia seleccionada.)

11.3 Preparación del Entorno -> Falta

- ***Entorno de Desarrollo****: para programación y pruebas unitarias.*
- ***Entorno de Pruebas/Staging****: réplica del entorno productivo para validaciones previas.*
- ***Entorno de Producción****: infraestructura final donde funcionará el sistema.*

Sección reservada

(Incluir especificaciones técnicas del servidor, sistema operativo, bases de datos, librerías y

configuraciones necesarias.)

11.4 Capacitación de Usuarios

- *Capacitación inicial para usuarios clave.*
- *Manual de usuario y guías rápidas.*
- *Sesiones prácticas con supervisión.*

11.5 Cronograma de Implementación

***Cronograma:** [Cronograma Angora]*

11.6 Plan de Contingencia

En caso de fallos durante la implementación del sistema Angora, se seguirán las siguientes acciones para mitigar riesgos y restaurar la estabilidad. Si se detecta un fallo crítico (e.g., errores en endpoints, fallos de autenticación con JWT, o problemas de despliegue), el equipo detendrá inmediatamente el despliegue y revertirá al último entorno estable utilizando un rollback en el repositorio GitHub, restaurando la rama main a la versión anterior mediante un git revert o git checkout a un commit funcional. Para el backend (angora.app), se restaurará la aplicación Spring Boot desde un backup del archivo WAR o la configuración del servidor (e.g., Tomcat), mientras que para el frontend se repondrá el bundle estático desde un backup en el servidor de hosting (e.g., Vercel). Se notificará a los usuarios mediante un mensaje genérico en la interfaz y se activará un entorno seguro de pruebas para analizar el fallo sin afectar producción. Además, se documentará el incidente, se identificará la causa raíz, y se implementará una corrección antes de un nuevo despliegue, asegurando que todas las

pruebas en Postman se realicen exitosamente antes de proceder.

12. Plan de Mantenimiento

Este capítulo detalla las actividades y procedimientos necesarios para garantizar que el sistema Angora continúe funcionando correctamente después de su implementación, corrigiendo fallos, mejorando funcionalidades y adaptándose a nuevas necesidades.

12.1 Objetivos del Mantenimiento

- *Mantener la operatividad del sistema a largo plazo.*
- *Corregir errores detectados tras la puesta en producción.*
- *Optimizar el rendimiento del sistema de acuerdo a la evolución tecnológica.*
- *Adaptar el sistema a cambios en la legislación, procesos internos o necesidades del negocio.*

12.2 Tipos de Mantenimiento

- ***Mantenimiento Correctivo**: reparación de fallos detectados en el sistema.*
- ***Mantenimiento Preventivo**: acciones planificadas para prevenir fallos futuros (actualizaciones, optimización de consultas, limpieza de logs).*
- ***Mantenimiento Perfectivo**: mejoras de rendimiento, usabilidad o nuevas funcionalidades solicitadas.*
- ***Mantenimiento Adaptativo**: modificaciones necesarias para ajustarse a cambios externos (nuevas normativas, cambios en sistemas externos).*

12.3 Procedimiento de Mantenimiento

El procedimiento de mantenimiento del sistema Angora sigue un flujo estructurado: primero, se detecta un problema mediante reportes de usuarios, monitoreo de logs en el backend (angora.app) o fallos identificados durante el uso del frontend, registrando detalles como fecha, módulo afectado y síntomas. En la etapa de análisis, el equipo revisa los logs, realiza pruebas en Postman para replicar el fallo en los endpoints y consulta el código en GitHub para identificar la causa raíz, documentando los hallazgos. Luego, en la planificación, se define una solución (e.g., corrección de un controlador o ajuste de estilos), se crea una rama de funcionalidad (e.g., feature/fix-issue) y se estima el tiempo necesario, coordinando con los involucrados. En la ejecución, se implementa el cambio en la rama, se realiza un commit con descripción clara y se solicita una revisión por pares antes de fusionar con main. Posteriormente, en las pruebas, se valida la corrección en un entorno de desarrollo con Postman, verificando que los endpoints funcionen y el frontend se comporte como esperado, corrigiendo errores si surgen. Finalmente, en el cierre, se despliega la actualización al entorno de producción, se notifica a los usuarios si aplica, se archiva la documentación del problema y la solución en el repositorio, y se cierra el ticket o reporte asociado.

12.4 Herramientas y Recursos para el Mantenimiento

El mantenimiento del sistema Angora se apoya en las siguientes herramientas y recursos: Git, utilizado para el control de versiones a través del repositorio en GitHub, permitiendo la gestión de ramas (e.g., feature/fix-issue) y el seguimiento de cambios con commits; Postman, empleado para pruebas y validación de endpoints durante la corrección de incidencias, asegurando que las APIs funcionen correctamente antes y después de los cambios; y Trello, usado para la gestión de incidencias y tareas, organizando los problemas detectados, asignando responsables y rastreando el progreso desde la detección hasta el cierre en tableros personalizados.

12.5 Frecuencia de Mantenimiento

- *Revisiones preventivas programadas cada **6** meses.*
- *Auditorías de seguridad cada **6** meses.*
- *Actualización de librerías y dependencias cada **4** meses o según parches críticos.*

12.6 Documentación y Registro de Cambios

*Todo cambio realizado al sistema debe registrarse en el **Historial de Versiones** con:*

- *Fecha del cambio.*
- *Descripción breve.*
- *Responsable.*
- *Versión del sistema.*

13. Glosario de Términos

Este glosario reúne las definiciones de términos técnicos, acrónimos y conceptos de negocio utilizados en el sistema Angora, facilitando la comprensión para desarrolladores, usuarios y personal de soporte.

13.1 Términos Técnicos

- ***API (Application Programming Interface)**: Conjunto de funciones y protocolos que permiten la comunicación entre sistemas.*

- ***Backend**: Parte del sistema que se ejecuta en el servidor, encargada de la lógica de negocio y el manejo de datos.*
- ***Frontend**: Interfaz gráfica del sistema que interactúa directamente con el usuario.*
- ***Base de Datos Relacional**: Estructura de almacenamiento de datos organizada en tablas con relaciones entre ellas.*
- ***Token de autenticación**: Código utilizado para validar y mantener la sesión de un usuario.*
- ***CRUD (Create, Read, Update, Delete)**: Operaciones básicas de manipulación de datos.*
- ***Deploy**: Proceso de poner una versión del sistema en un entorno específico (desarrollo, pruebas o producción).*
- ***Endpoint**: URL específica en una API para acceder a un recurso o ejecutar una acción.*
- ***SPA (Single Page Application)**: Aplicación web que carga una sola página HTML y actualiza el contenido dinámicamente sin recargar.*

13.2 Términos de Negocio

- ***Factura**: Documento que respalda una transacción comercial entre la empresa y un cliente.*
- ***Inventario**: Conjunto de productos, materias primas y recursos almacenados.*
- ***Orden de Compra**: Documento que especifica productos o servicios solicitados a un proveedor.*
- ***Materia Prima**: Insumo utilizado para la fabricación o producción de bienes.*
- ***Proveedor**: Persona o empresa que suministra productos o servicios a la organización.*
- ***Pedido**: Solicitud de productos o servicios realizada por un cliente.*
- ***Cliente**: Persona o entidad que adquiere los bienes o servicios ofrecidos por la empresa.*

- ***Usuario Clave**: Persona dentro de la empresa con alto conocimiento del sistema y procesos, encargada de la capacitación y validación.*

13.3 Acrónimos Usados en el Proyecto

***ANGORA**: Acrónimo del nombre del proyecto, que representa el sistema principal de la empresa productora y comercializadora de productos para el hogar Fraganc'eys, con el siguiente significado:*

- ***A** - Automatización de procesos: Facilita la gestión automatizada de la producción y ventas.*
- ***N** - Navegación intuitiva: Ofrece una interfaz fácil de usar para los empleados y el administrador.*
- ***G** - Gestión de procesos: Optimiza el control de inventarios, facturas, clientes y pedidos.*
- ***O** - Optimización de tiempos: Reduce tiempos en operaciones diarias logísticas y financieras.*
- ***R** – Rendimiento optimo: Ofrece respuestas rápidas y optimizadas, haciendo los procesos casi de forma instantánea.*
- ***A** - Análisis de información: Proporciona datos clave para decisiones estratégicas.*

  Este acrónimo encapsula las fortalezas del sistema, diseñado para hacer la vida en el hogar más eficiente y organizada.

14. Control de Cambios

Este capítulo tiene como objetivo mantener un historial detallado de todas las modificaciones realizadas al sistema Angora, permitiendo rastrear su evolución y garantizar la trazabilidad de los cambios a lo largo de su ciclo de vida.

14.1 Propósito del Control de Cambios

- *Documentar modificaciones de código, configuraciones y arquitectura.*
- *Facilitar auditorías y revisiones técnicas.*
- *Mantener la coherencia entre el código, la documentación y el sistema en producción.*
- *Evitar la pérdida de información y cambios no autorizados.*

14.4 Herramientas de Control de Versiones

- ***Git** como sistema de control de versiones distribuido.*

  ***GitHub** para repositorio remoto y gestión de ramas.*
- *Convenciones de branching:*
  - `main` → *Rama estable y en producción.*
  - `feature/*` → *Nuevas funcionalidades.*
  - `fixed` → *Corrección de algunos errores.*

15. Anexos

Este capítulo recopila información adicional que complementa el manual técnico de Angora. Los anexos pueden incluir diagramas, capturas de pantalla, configuraciones, listados de código y cualquier otro material que facilite la comprensión del sistema.

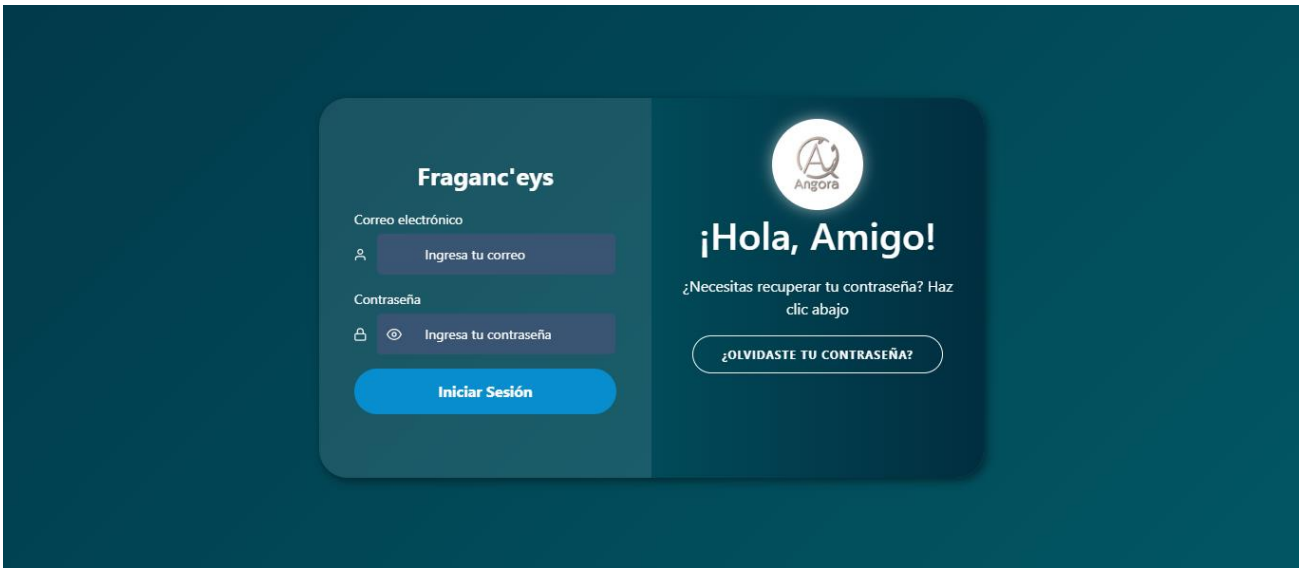
15.2 Capturas de Pantalla

Sección reservada

(Colocar imágenes que muestren las interfaces del sistema, pantallas clave como login,

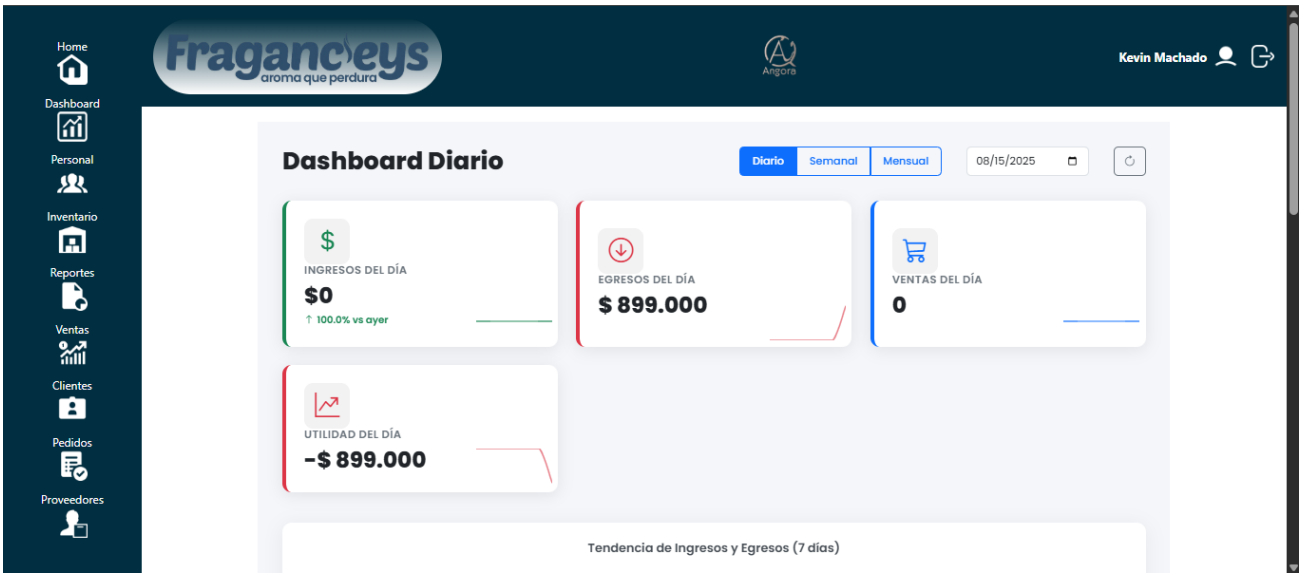
dashboard, gestión de pedidos, etc.)

1. ***Login***

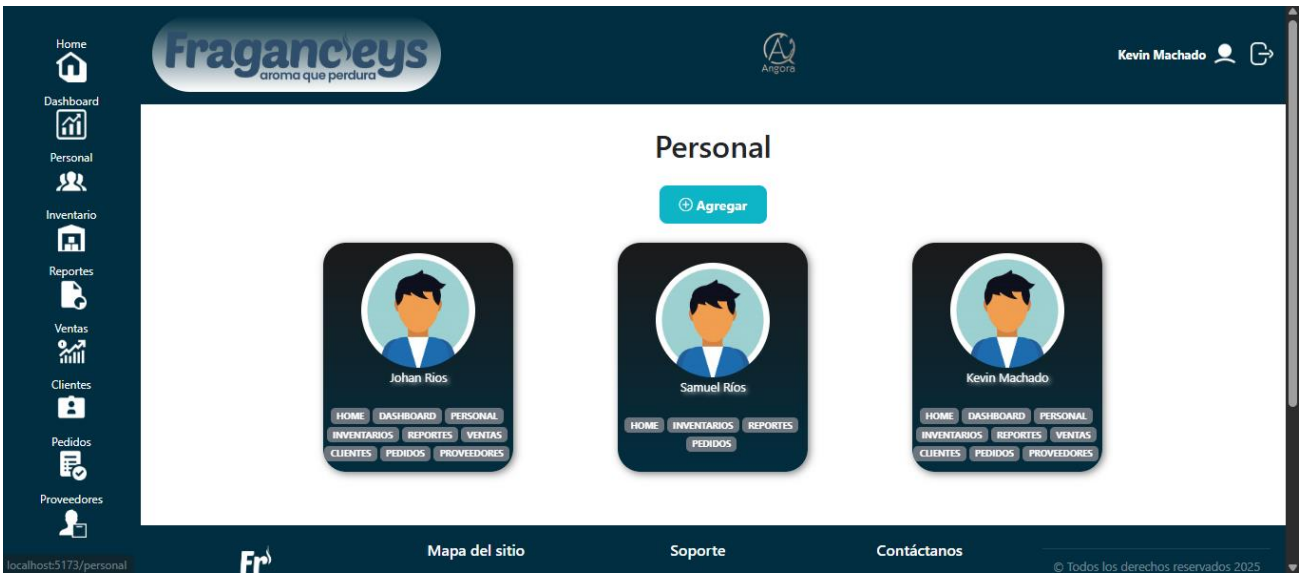


2. ***Home***

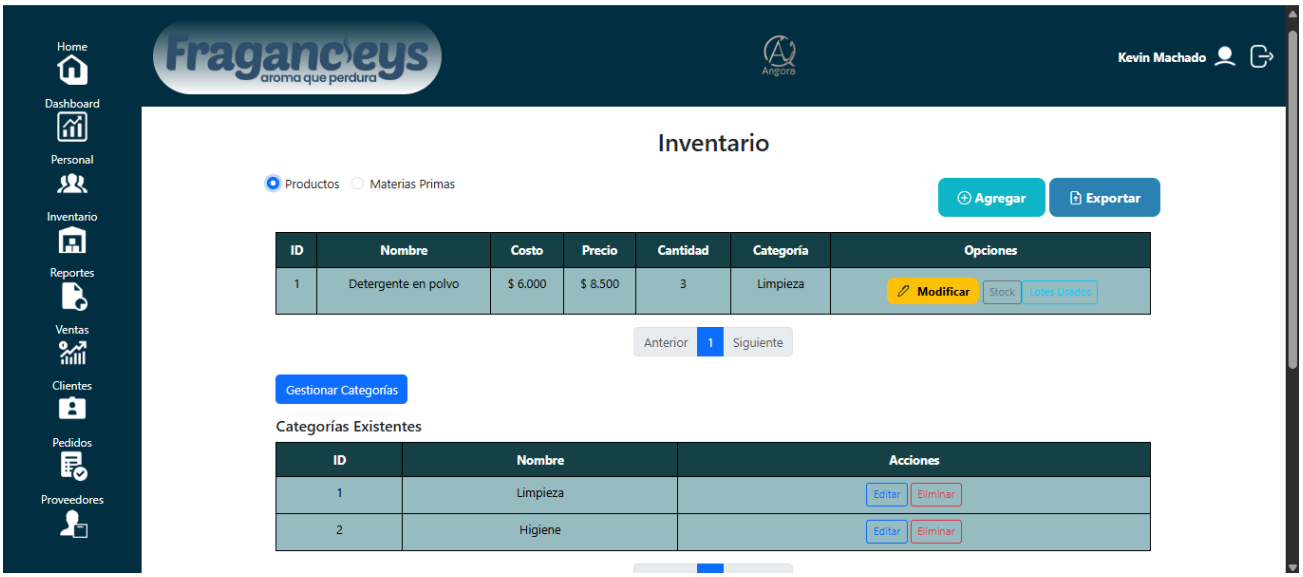
3. *Dashboard*



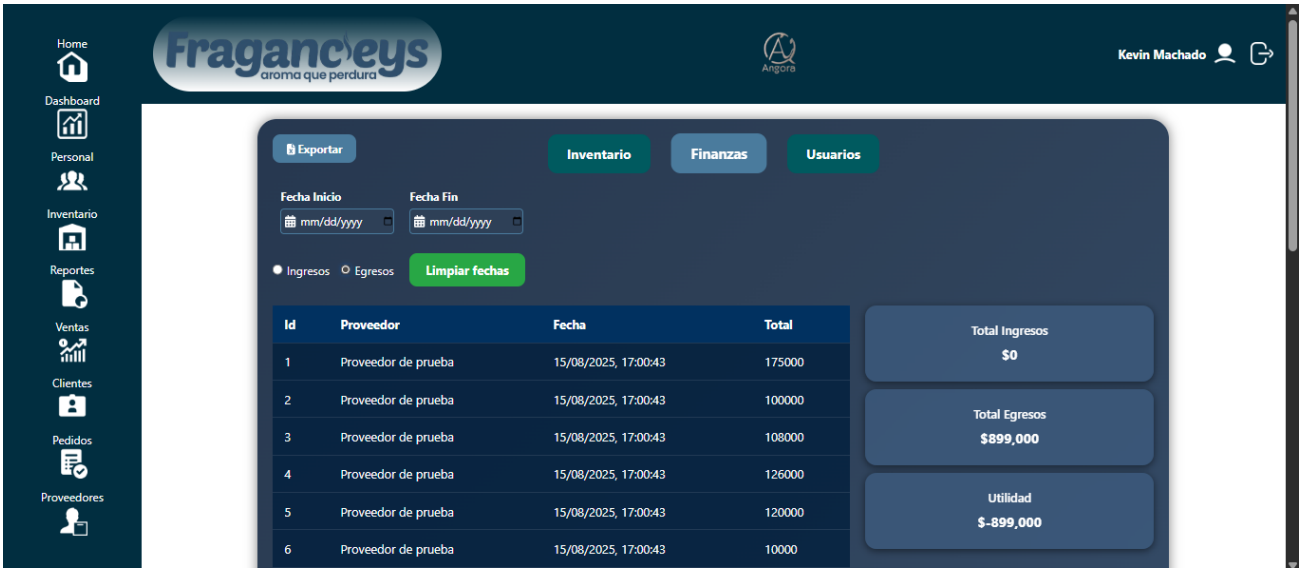
4. *Personal*



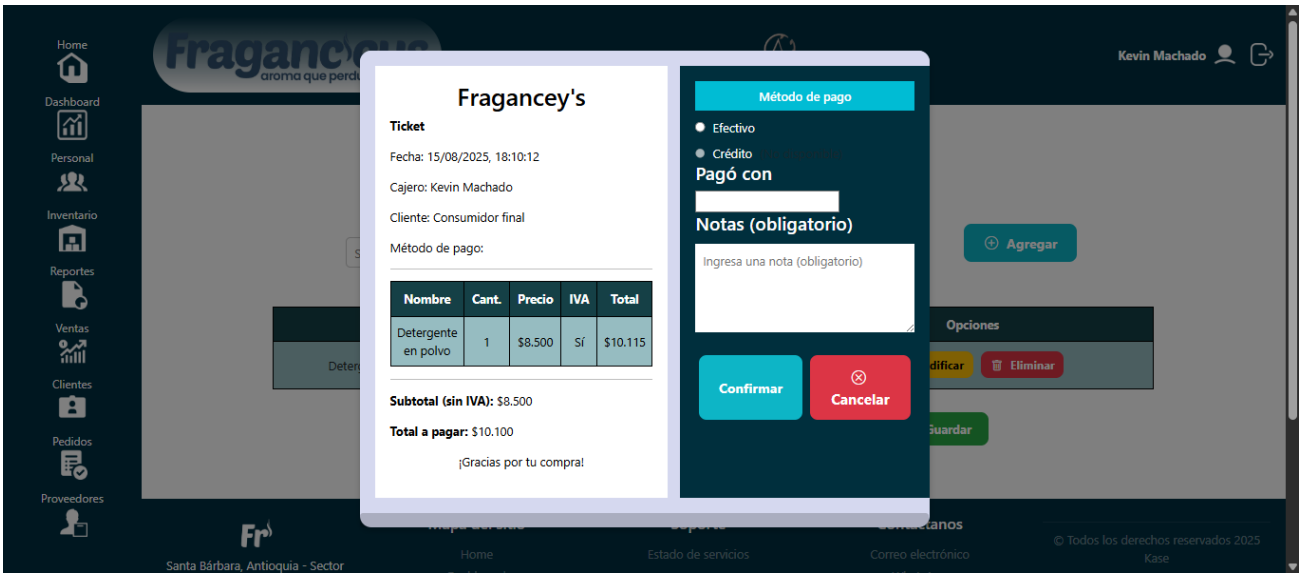
5. *Inventario*



6. *Reportes*

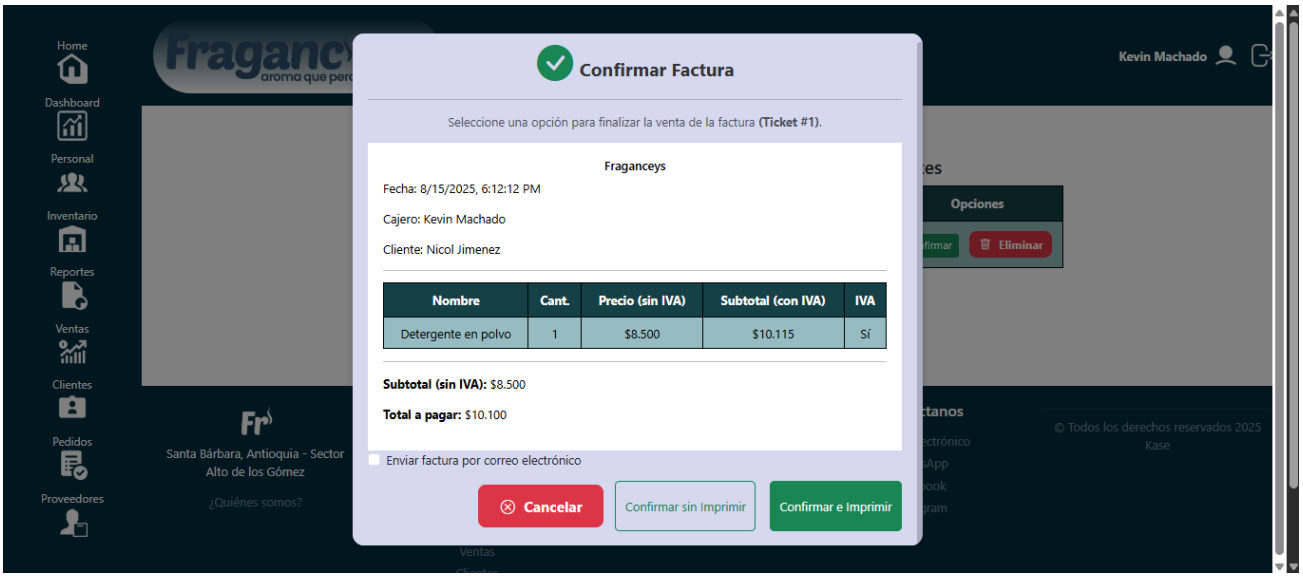


7. *Ventas*



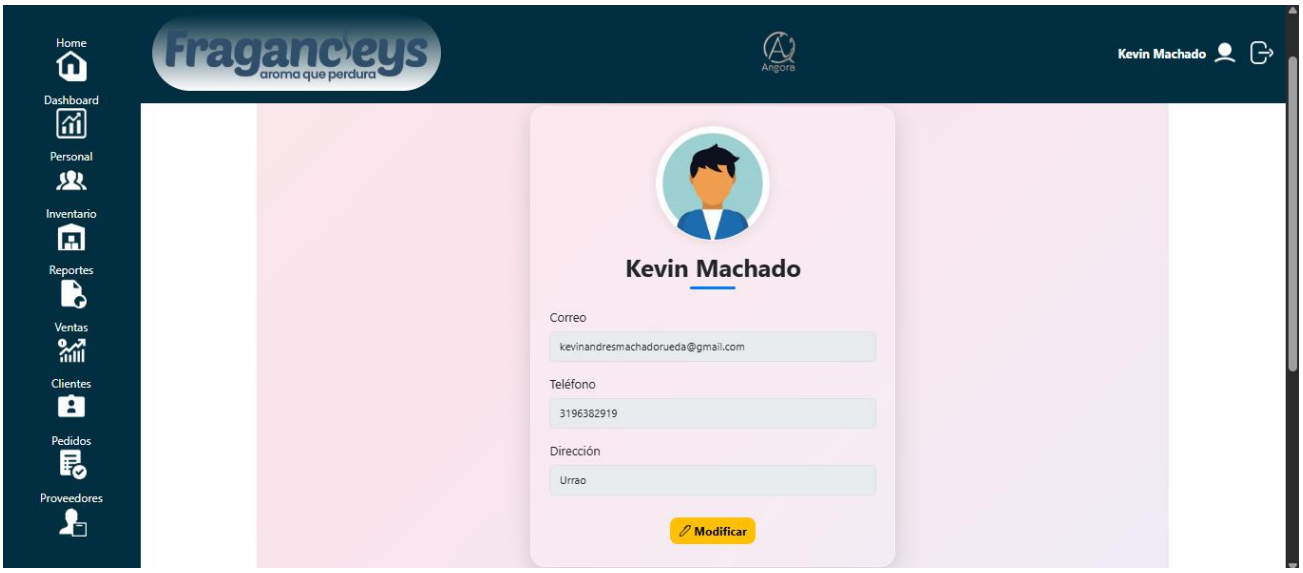
8. *Clientes*

9. Pedidos



10. Proveedores

11. Perfil



15.3 Configuraciones de Servidor y Entorno

***Entorno de Desarrollo**:*

- ***Sistema Operativo**: Windows 10 o 11 para desarrollo local.*
- ***Versión de Java**: Java 21 con Spring Boot 3.5.3.*
- ***Base de Datos**: MySQL con WorkBench 8.0.40, configurada localmente con un esquema angora_db.*
- ***Puertos**: Backend en puerto 8080, frontend en puerto 5173 (React dev server).*
- ***Notas**: Utiliza un servidor Tomcat embebido en Spring Boot para pruebas locales.*

***Entorno de Pruebas**:*

- **Falta**

***Entorno de Producción**:*

- **Falta**